

****

**J I A N G S U U N I V E R S I T Y**

**软件项目开发实践课程设计报告**

|  |  |
| --- | --- |
| 学院名称： | 计算机科学与通信工程学院 |
| 专业班级： | 软件工程17级 |
| 学生姓名： | 刘祚滔 |
| 学生学号： | 3171304030 |
| 指导教师： | 陈继明老师 |

**2020年6月27日**

目录

[第一章 课程设计概述 3](#_Toc44664334)

[第二章 系统整体设计 4](#_Toc44664335)

[第三章 模块的设计和实现 4](#_Toc44664336)

[3.1 数据库设计与实现 4](#_Toc44664337)

[3.2 连接访问数据库操作（model层+dao层+service层） 6](#_Toc44664338)

[3.3 View（前端视图） 9](#_Toc44664339)

[3.4 Controller（控制器） 9](#_Toc44664340)

[第四章 运行测试和心得总结 9](#_Toc44664341)

[4.1运行测试 9](#_Toc44664342)

[4.2心得总结 9](#_Toc44664343)

## 第一章 课程设计概述

### 1.1设计目的

开发出一个课程管理系统，可以实现基本的查询、评论功能。

### 1.2功能要求

1.进入系统之后，能够看到课程列表（如图1.2.1）。

2.课程列表页面通过课程名称模糊查询到课程。

3.课程列表页面选择课程点击“查看章节”跳转到章节列表（图1.2.2）。

4.章节列表页面显示出课程名称、难度等级和章节列表内容。

5.课程列表页面选择课程点击“查看评论”跳转到评论列表页面（如图三）。

6.评论列表页面显示出课程名称、难度等级、综合评论星级和评论列表内容

7.评论列表页签提供发表评论入口。星级下拉框为1星到5星。

8.点击“发表”使用AJAX提交内容，如果提交成功则刷新当前页面，评论列表当中显示新追加的评论信息。

9.课程的评论星级为该课程所有评论的星级平均值。

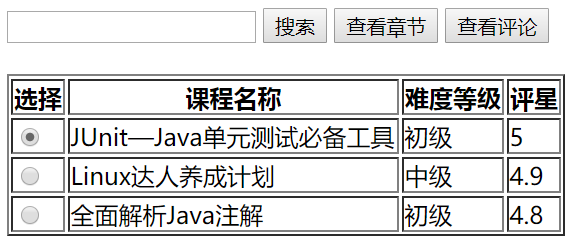


图1.2.1

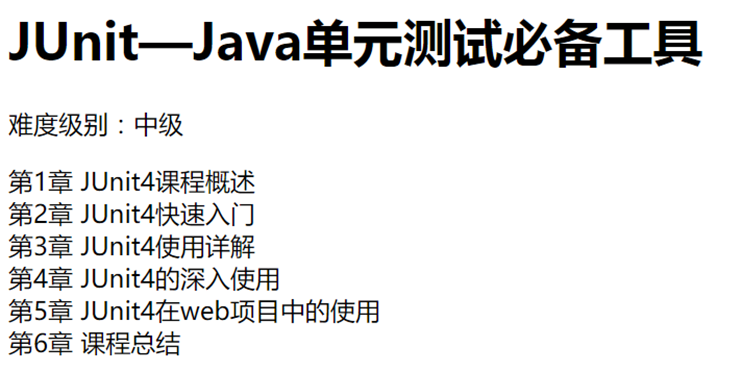


图1.2.2

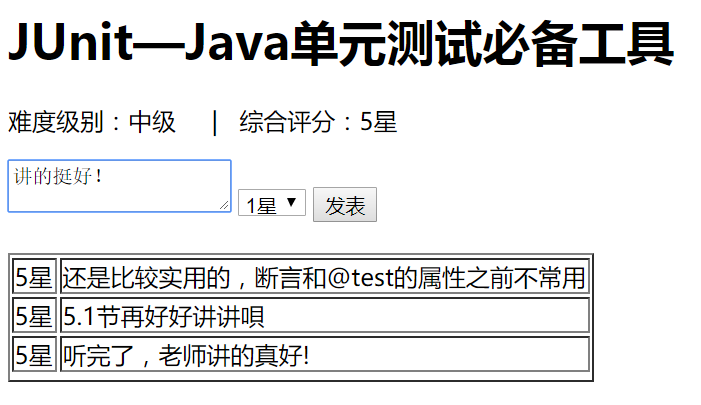


图1.2.3

### 1.3开发环境

电脑系统：Windows 10

JDK版本：Java version “1.8.0\_251

JAVA集成开发环境：IDEA 2020.1.2

项目管理工具：Maven-3.6.3

后端使用框架：Spring Boot 2.3.1

前端使用框架：Thymeleaf

数据库：Navicat+Mysql8.0

连接数据库：Mybatis1.3.2

## 系统整体设计

系统使用MVC模式开发，分为DAO层，Service层，Controller层、View层。

DAO层：DAO层主要是做数据持久层的工作，负责与数据库进行联络的一些任务都封装在此，通过mybatis的配置文件.xml，可在模块中调用此接口来进行数据业务的处理，而不用关心此接口的具体实现类是哪个类。

Service层：Service层主要负责业务模块的逻辑应用设计。同样是首先设计接口，再设计其实现的类，接着再Spring的配置文件中配置其实现的关联。这样我们就可以在应用中调用Service接口来进行业务处理。封装Service层的业务逻辑有利于通用的业务逻辑的独立性和重复利用性。实现定义出来的功能模块的功能。Service层是建立在DAO层之上的，建立了DAO层后才可以建立Service层，而Service层又是在Controller层之下的，因而Service层应该既调用DAO层的接口，又要提供接口给Controller层的类来进行调用，它刚好处于一个中间层的位置。每个模型都有一个Service接口，每个接口分别封装各自的业务处理方法。

Controller层：Controller层负责具体的业务模块流程的控制，在此层里面要调用Serice层的接口来控制业务流程，控制的配置也同样是在Spring的配置文件里面进行，针对具体的业务流程，会有不同的控制器。

View层即负责前台页面的显示，与Controller结合紧密。

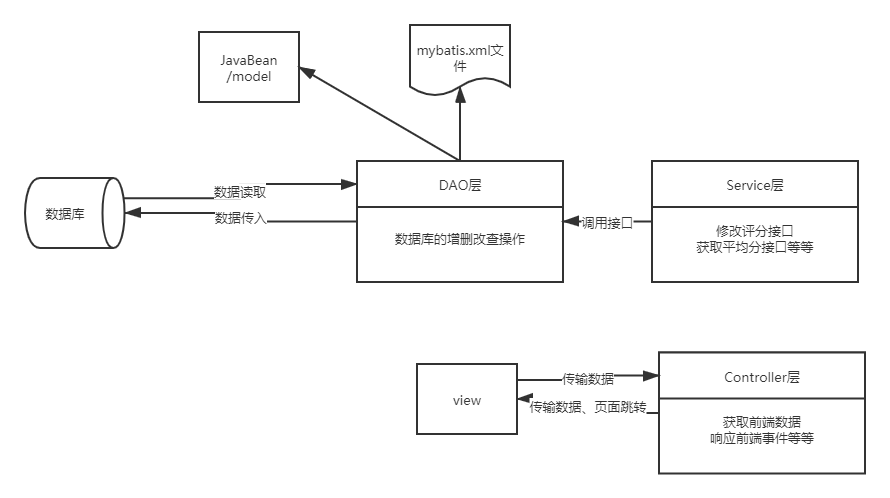
数据流图如下图所示：  


图2.1

## 第三章 模块的设计和实现

### 3.1 数据库设计与实现

本系统需要的存储的相关信息有课程的星级信息、难度信息、章节信息、以及评论信息等，所以数据库对应建立了三个表：课程基本信息表：course、课程章节信息表：chapter以及课程评论信息表comment

#### 3.1.1课程基本信息表：course

课程基本信息有课程名称、难度等级以及课程星级，此外，为了实现课程基本信息与课程章节信息以及课程评论信息的一对多关系，建成一个表的话明显会造成冗余信息过多，所以应该将这些信息分开，课程章节信息以及课程评论信息两个表应当添加一个外键用来实现与课程基本信息表的关联，所以我们新添加了一个自增的id作为课程的唯一标识与主键，同时作为其他表的外键实现表之间的关联。

数据字典如表一所示：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名 | 类型 | 长度 | 是否主键 | 是否可为空 | 注释 |
| Id | Bight | 10 | 是 | 否 | 课程唯一id |
| course\_name | Varchar | 20 | 否 | 否 | 课程名称 |
| difficulty\_level | Varchar | 20 | 否 | 否 | 课程难度等级 |
| star\_level | Varchar | 10 | 否 | 否 | 课程评星 |

表3.1.1.1：course数据字典

数据库图像如图一表示：



图3.1.1.2：course数据库图像

#### 3.1.2 课程章节信息表：chapter

为了防止造成过多冗余，将课程基本信息表的主键id作为外键引入表，其余则为课程章节的相关信息。

数据字典如表二所示：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名 | 类型 | 长度 | 是否主键 | 是否可为空 | 注释 |
| id | Bight | 10 | 否 | 否 | 对应课程的唯一标识 |
| chapter\_num | Int | 10 | 否 | 否 | 章节标识 |
| chapter\_info | varchar | 50 | 否 | 否 | 章节信息 |

表3.1.2.1：chapter数据字典

数据库图像如图一表示：



图3.1.2.2：chapter数据库图像

#### 3.1.3课程评论信息表comment

为了防止造成过多冗余，将课程基本信息表的主键id作为外键引入表，其余为课程评论的基本信息。

数据字典如表三所示：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性名 | 类型 | 长度 | 是否主键 | 是否可为空 | 注释 |
| Id | Bight | 10 | 否 | 否 | 对应课程的唯一标识 |
| star\_level | Varchar | 10 | 否 | 否 | 评论星级 |
| comment\_info | Varchar | 50 | 否 | 否 | 评论内容 |

表3.1.3.1：comment数据库字典

数据库图像如图一表示：

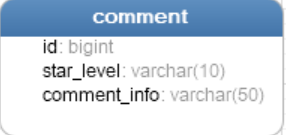
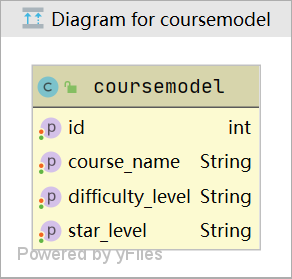
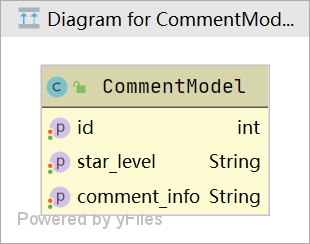


图3.1.3.2：comment数据库图像

### 3.2 连接访问数据库操作（model层+dao层+service层）

#### 3.2.1创建数据库表对应的实体类（model层）

创建数据库表对应的JavaBean用于临时存储与传递数据库读出的数据，下图为相应的类图，省略了get/set方法。

图五：课程基本信息实体类图 图六：评论实体类图

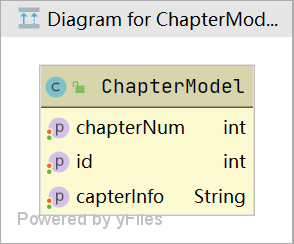


图3.2.1.1：课程章节实体类图

#### 3.2.2对数据库进行数据持久化操作（dao层）

实现对数据库的持久化操作，实现一些基本的增删改查，这里使用了mybatis，将server方法与创建的xml文件进行一一映射，实现自动注入，只需写出mapper接口即可。下面的类图表示相应的对应数据库表的mapper接口：

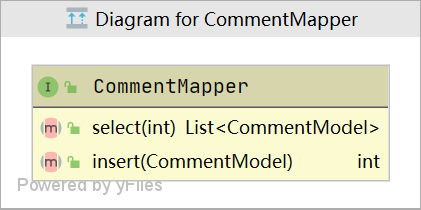
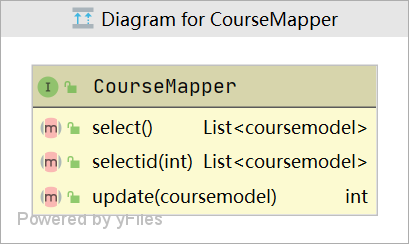


图3.2.2.1：课程基本信息mapper类图 图3.2.2.2：评论内容mapper类图

课程基本信息mapper函数说明：

|  |  |
| --- | --- |
| Select() | 查询所有课程信息 |
| selectid(int id) | 根据id查询课程信息 |
| update(coursemodel cml) | 根据传入的课程数据修改相应课程数据，主要目的是根据评论修改课程评分 |

评论内容mapper函数说明

|  |  |
| --- | --- |
| select(int id) | 根据课程id查询评论信息 |
| insert(CommentModel commentModel) | 添加评论信息 |

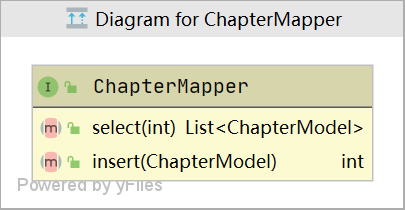


图3.2.2.3：课程章节mapper类图

课程章节mapper函数说明：

|  |  |
| --- | --- |
| select(int id) | 根据id查询课程章节集合 |
| insert(ChapterModel chapterModel) | 插入章节(实际上没有必要插入章节) |

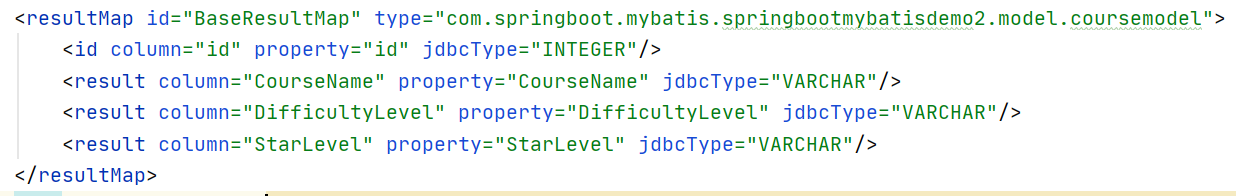
Mybatis配置xml文件（以course课程查询为例）

一：配置mapper路径包



二：写入结果映射

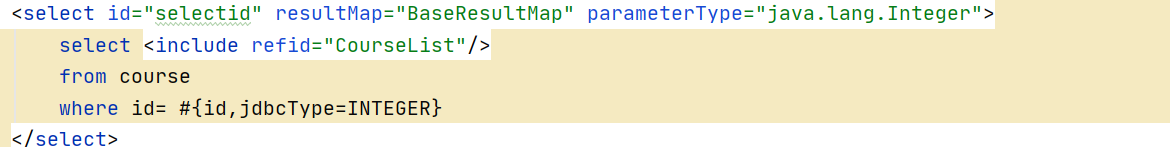
resultMap 元素是 MyBatis中最重要最强大的元素，将返回的结果集注入到dao层的javabean中，需要配置到相应的dao路径并配置对应属性



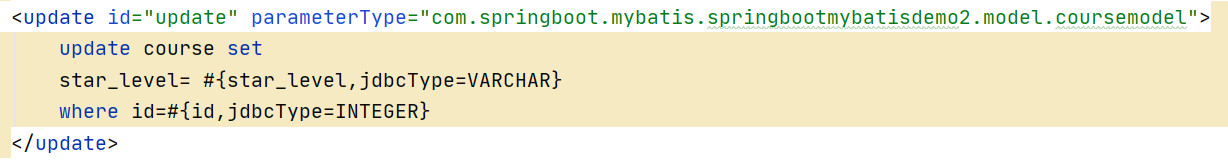
三：相应的数据库增删改查映射

Resultmap设定返回的结果映射、parameterType设定输入的数据类型

1:根据id查询信息



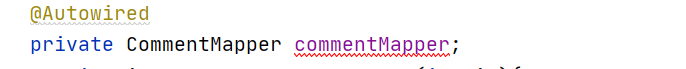
2：修改评分



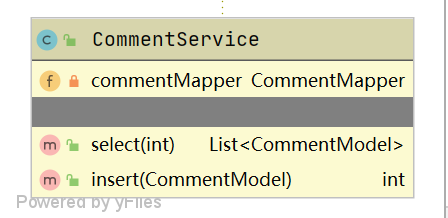
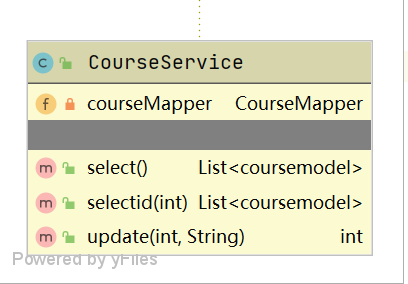
#### 3.2.3 实现具体的功能的类（service层）

这里封装了具体功能供给控制层调用，将系统需要的对数据库的操作封装成类，而不再是直接的增删改查。

这里可以通过依赖注入直接将Mapper对象注解进入

例：

对课程信息操作的具体功能类：CourseService,图十为该类类图：

图3.2.3.1：CourseService类图 图3.2.3.2：CommentService类图

在该类中，select()与select(id)都是直接调用mapper编写的方法，而update主要是为了修改评分而不是为了修改其他数据，所以这里只输入了课程id与要修改的评分数据：该类流程图如图12所示：

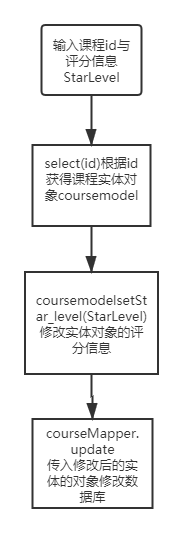
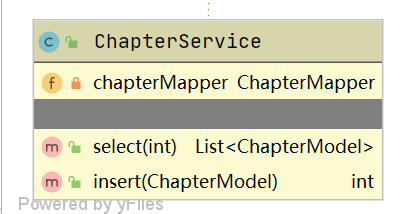
 

图3.2.3.3：update流程图 图3.2.3.4：ChapterService类图

其余service均为直接调用mapper增删改查函数，因为是新用springboot框架，所以没有将service的功能分详细，实际上课程评论对应的服务应该还可以添加求取评论星级平均数等等功能，但是后来直接写到控制层了，这是不对的。

### 3.3 View（前端视图）

视图层使用thymeleaf框架编写，因为springboot对thymeleaf的支持度比较好，同时thymeleaf的语法不会破坏html的结构。

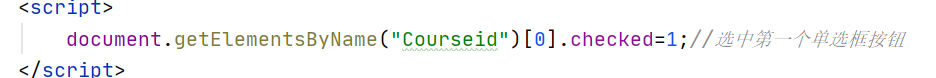
#### 3.3.1 课程列表显示界面（CourseList.html）

要显示课程列表主要是对后端发送的数据的循环读取，thymeleaf支持each迭代循环，依次从传入的数据集中读取数据到列表



图3.3.1.1：each循环读取数据

在each循环中为每行添加了单选按钮Courseid保存课程的唯一id，同时在列表加载结束之后将id为1的按钮设置.checked为1,默认设置选中第一行。



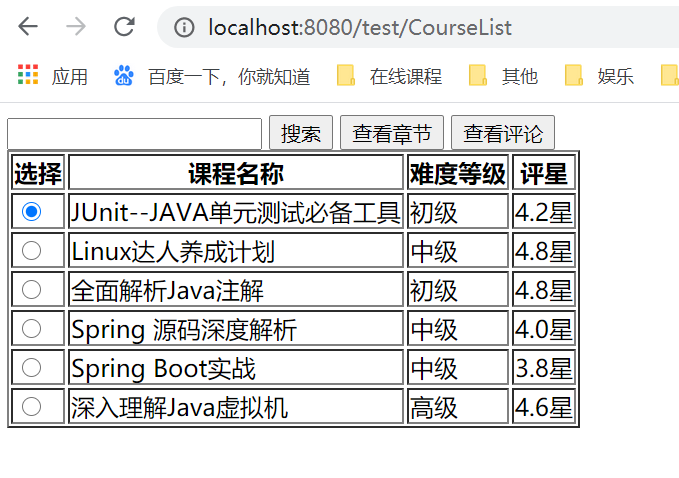


图3.3.1.2：课程列表页面图

#### 3.3.2 课程章节列表页面（Chapter.html）

后台传入的数据有课程基本信息以及课程章节信息，课程基本信息表示在标题出。each循环显示章节列表

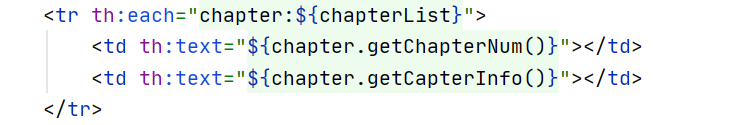
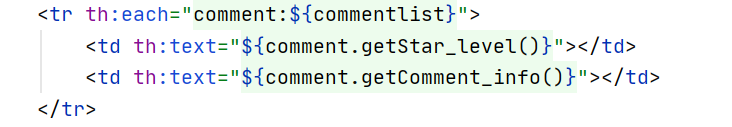


图3.3.2.1：课程章节列表页面图

#### 3.3.3课程评论列表(Comment.html)

前端传入课程相关信息以及章节信息，each循环显示评论信息在列表：



使用Ajax实现在页面不跳转的情况下与后端交互数据：

Ajax配置如图16：



图3.3.2.2：Ajax配置图

使用Json格式作为数据传输，首先建立Javascriot对象，因为Json支持直接将Javascript对象转换为Json字符串格式

要传输的数据对象：



Id为课程id，starLevel为评价星级，cmtInfo为评价内容

Ajax发送数据流程图如图17所示：

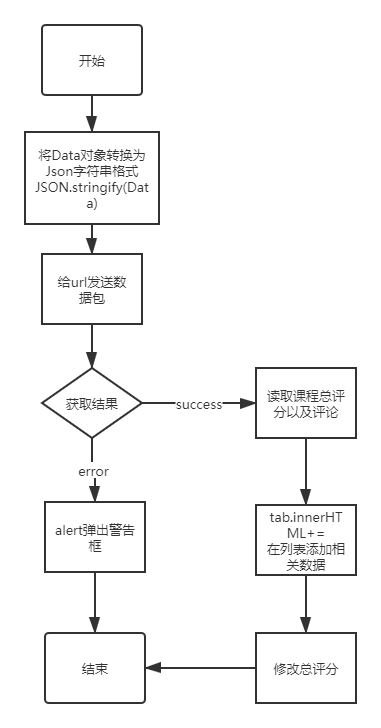
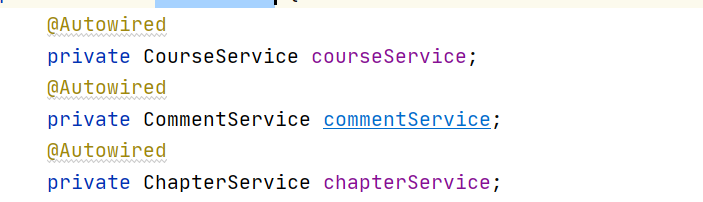


图3.3.3.1 ajax发送接收数据流程图 图3.3.3.2：课程评论界面

### 3.4 Controller（控制器）

这里控制器我只写了一个类CourseShow来控制页面，依赖注入服务对象



#### 3.4.1 实现根据书名模糊查询

原理是将查询的内容与书名匹配，只要有部分字符串匹配即加入到结果集之中。流程图如图19表示：

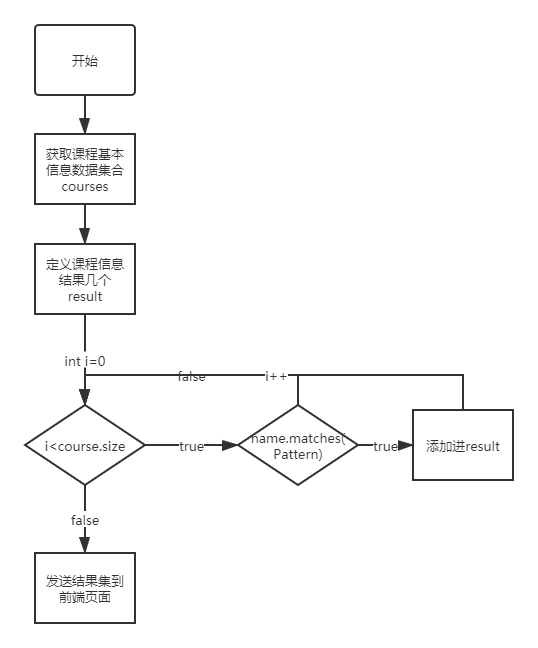


图3.4.1.1：实现书名模糊查询

#### 3.4.2 Ajax接收评论数据

在前端使用Ajax发送数据后，控制器应定义一个路径用来接收Ajax发送的数据并进行处理返回需要的数据。使用@RequestMapping("/recrivecomment")定义路径，而后定义返回信息为@ResponseBody而不是跳转网页，因为Ajax前面使用的是post传输，所以使用@RequestBody String data定义数据参数，结构如图20所示：

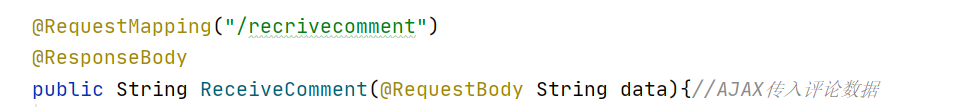


图3.4.2.1：接收数据函数结构

从接收数据到发送数据流程图如图21所示：

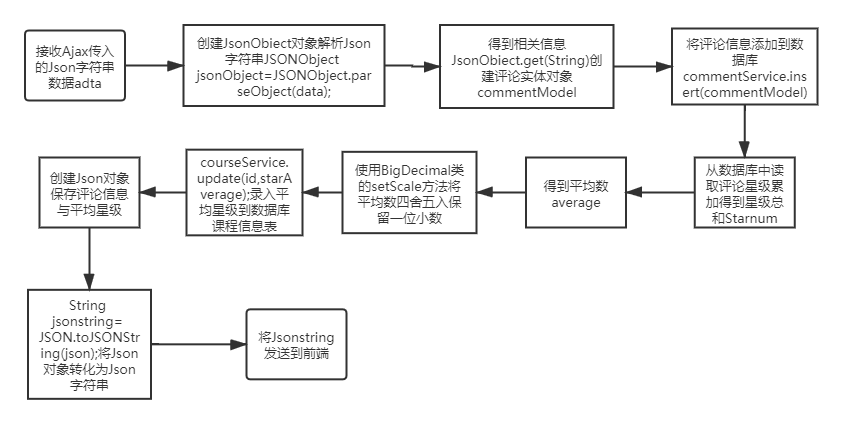
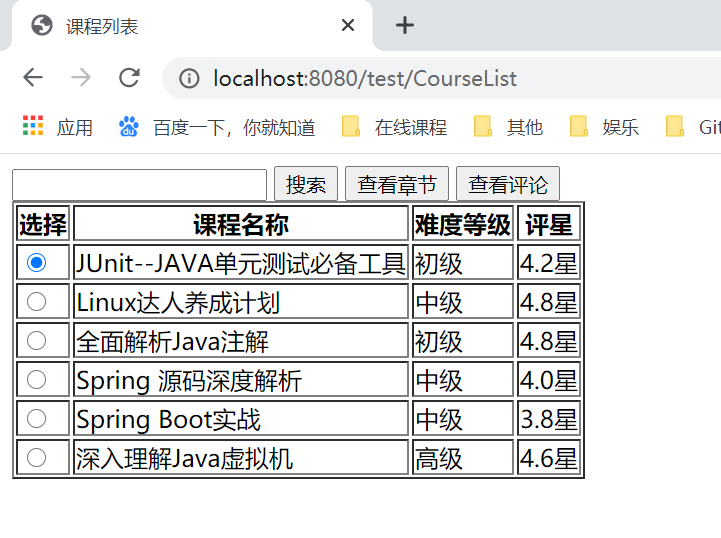


图3.4.2.2：后端接收Ajax发送的数据并返回流程图

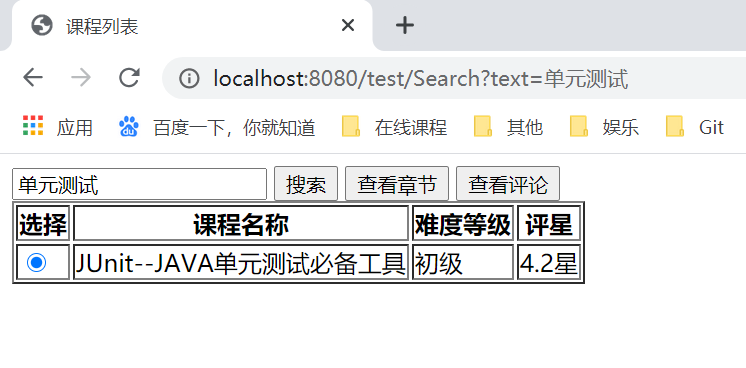
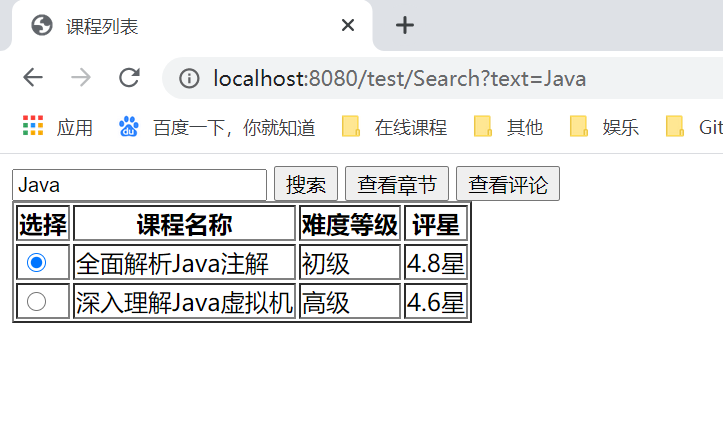
## 第四章 运行测试和心得总结

### 4.1运行测试

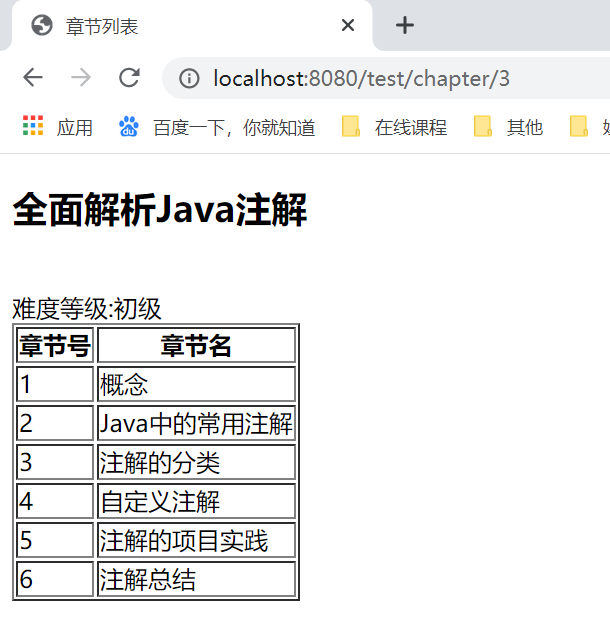
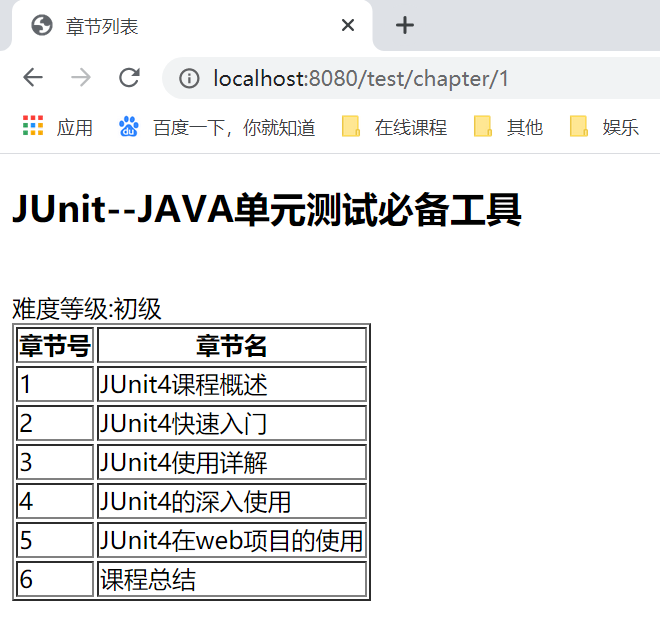
1.进入系统之后，能够看到课程列表。



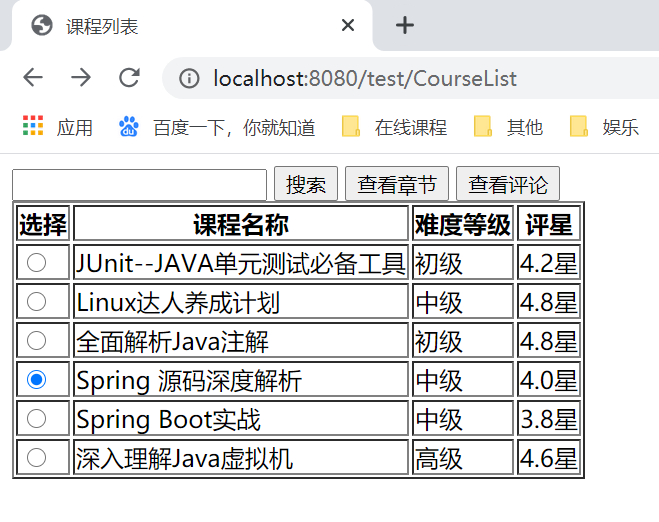
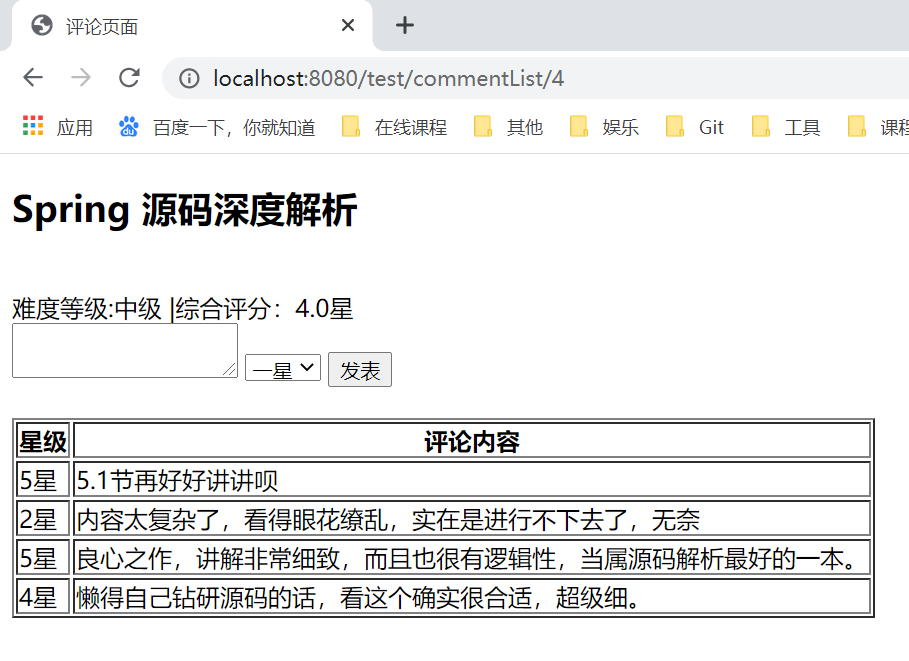
2.课程列表页面通过课程名称模糊查询到课程。

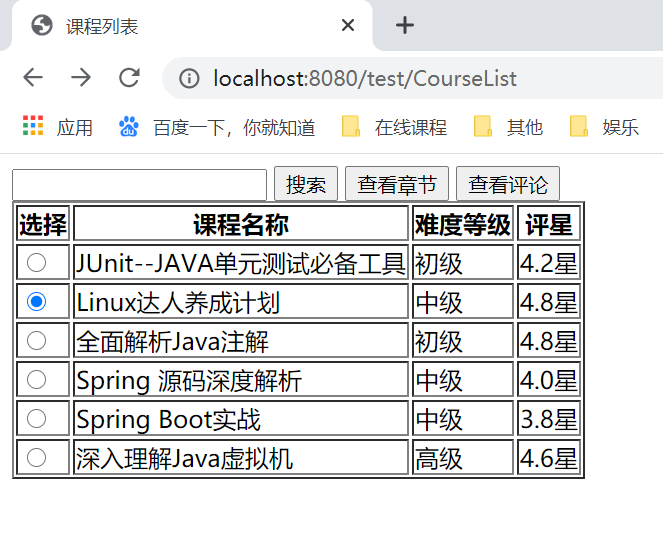
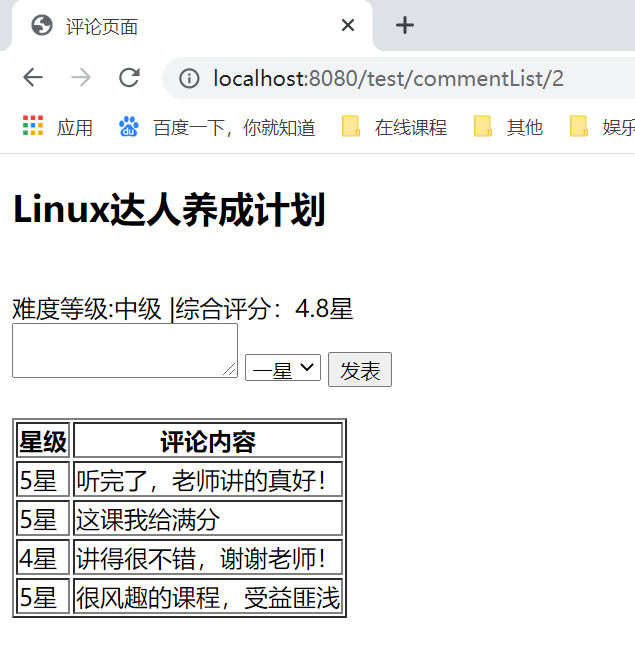


3.课程列表页面选择课程点击“查看章节”跳转到章节列表。章节列表页面显示出课程名称、难度等级和章节列表内容。

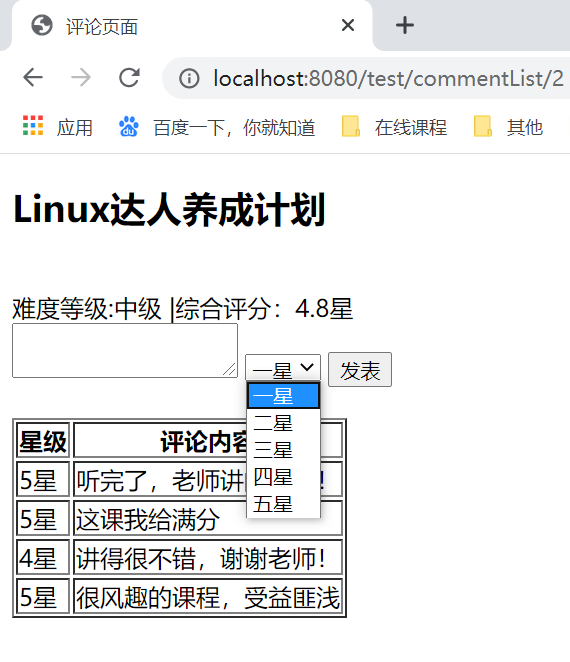


5.课程列表页面选择课程点击“查看评论”跳转到评论列表页面。评论列表页面显示出课程名称、难度等级、综合评论星级和评论列表内容

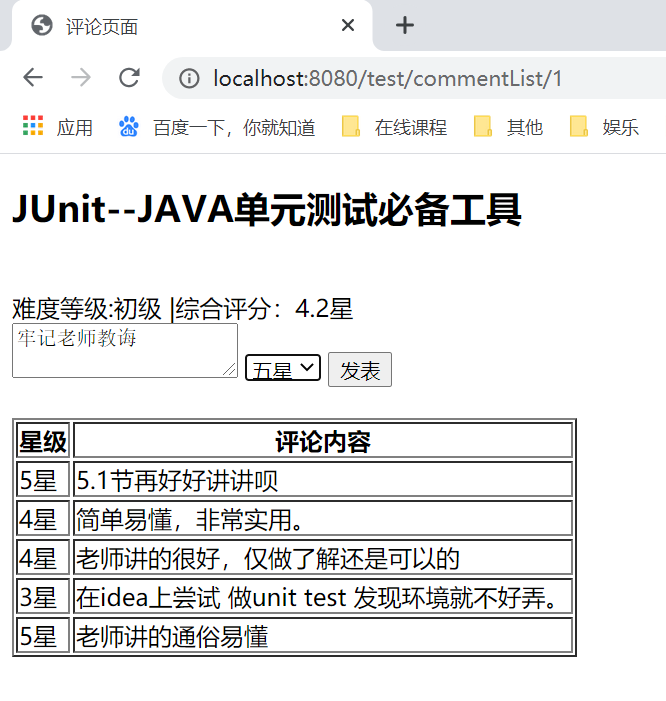
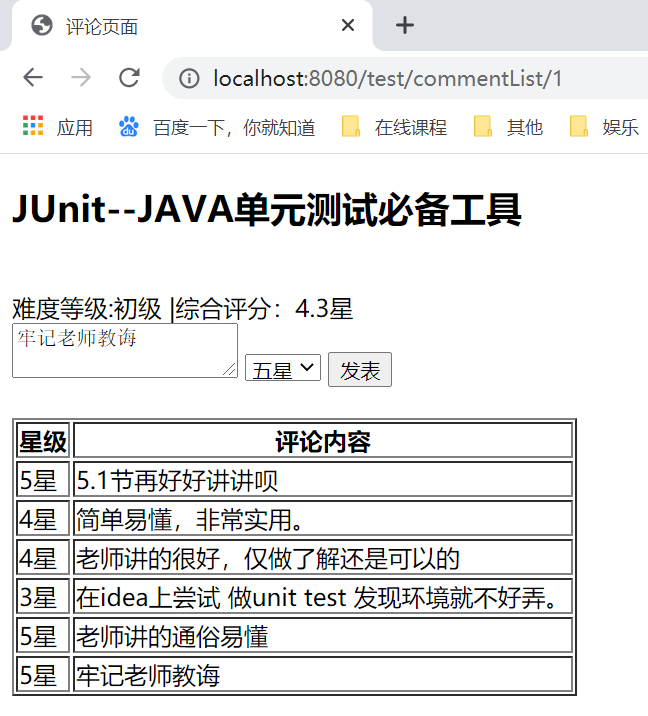
 

7.评论列表页签提供发表评论入口。星级下拉框为1星到5星。



8.点击“发表”使用AJAX提交内容，如果提交成功则刷新当前页面，评论列表当中显示新追加的评论信息。课程的评论星级为该课程所有评论的星级平均值。

### 4.2心得总结

在这个系统开发过程中，我新学习了springboot、mybatis等等以前没有用过的框架与技术，所以在开发过程中遇到了很多问题，比如没有将service的功能分详细，实际上课程评论对应的服务应该还可以添加求取评论星级平均数等等功能，但是后来直接写到控制层了，这是不对的。不过再回过头来写报告的时候就发现了这些问题，当时在写程序的时候是没有意识到的，没有深入思考为什么他要这样用。这次课程管理系统开发我学到了很多东西，受益匪浅。